ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
UDC/UDK 004.832.021.023:519.83

A NOVEL ITERATIVE OPTIMIZATION ALGORITHM BASED ON DYNAMIC RANDOM POPULATION

Seyyed Meysam Hosseini, Hamid Reza Mirsalari, Hossein Pourhoudhiary

Original scientific paper
Various heuristic optimization methods have been developed in artificial intelligence. These methods are mostly inspired by natural evolution or some applicable innovations, which seek good (near-optimal) solutions at a reasonable computational cost for search problems. A new iterative optimization algorithm is proposed in this paper. The algorithm is based on searching the most valuable part of the solution space, which is normally concentrated about a targeted bias vector (in the form of a dynamic random population). This algorithm greedily searches the solution space for global extremum. The comparison results between the proposed algorithm and some of the well-known heuristic search methods confirm the superiority of our proposed method in solving various non-linear optimization problems from the viewpoint of simplicity and accuracy.

Keywords: *dynamic random population, heuristic search algorithm, optimization*

Novi iterativni algoritam optimalizacije zasnovan na dinamičnoj slučajnoj populaciji

Izvorni znanstveni članak
U umjetnoj inteligenciji razvijene su različite heurističke metode optimalizacije. Te su metode uglavnom potaknute prirodnom evolucijom ili nekim primjenljivim inovacijama koje traže dobra (gotovo optimalna) rješenja uz razumnu računalnu cijenu za istraživane probleme. U radu se predlaže novi iterativni algoritam optimalizacije. Algoritam se zasniva na pretraživanju najvrednijeg dijela područja rješenja, koje je uobičajeno koncentrirano oko ciljanog (bias) vektora (u obliku dinamične slučajne populacije). Taj algoritam nezasitno pretražuje prostor rješenja u potrazi za globalnim ekstremom. Usporedba rezultata predloženog algoritma i nekih poznatih heurističkih metoda pretraživanja potvrđuje superiornost naše predložene metode u rješavanju različitih nelinearnih problema optimalizacije sa stajališta jednostavnosti i točnosti.

Ključne riječi: *dinamična slučajna populacija, heuristički algoritam pretraživanja, optimalizacija*

1 Introduction

Nowadays, the optimum use of time, energy and resources is of a great importance. Hence, many things can be seen from the perspective of optimization. Optimization problems almost exist in many fields of science, engineering and business for optimal use of facilities. But, since many of these optimization problems are inherently complex more efficient tools for solving them have always been considered. Furthermore, in an optimization problem with a high dimensional search space classical algorithms do not lead to the optimum solution, because search space increases exponentially with the problem size. Therefore, solving such problems with classical methods is not practical [1]. In the past, intelligent techniques for solving these problems have been presented that are mostly inspired by natural phenomena. Genetic algorithm, simulated annealing, ant colony search algorithm, particle swarm optimization and gravitational search algorithm are examples of these techniques [2, 3, 4, 5]. The important point regarding these methods is that their performance has a high dependency on the nature of the problems. In better words, the performance of different approaches varies when dealing with various problems and there is almost no simple method that can achieve the best solution for all problems.

In this paper, a novel iterative optimization algorithm is proposed based on search using dynamic random population with normal distribution. In fact, in this algorithm the solution space is searched based on dynamic random population with a special statistical distribution under a particular focus on the most valuable part of the solution space.

This paper is organized as follows. Section 2 provides a brief review of some existing heuristics algorithms. In Section 3 formulation of an optimization problem is described and the proposed algorithm and its characteristics are considered. Our comparative study and experimental results are demonstrated in Section 4 and finally some conclusion remarks are drawn in Section 5.

2 A brief review of some existing heuristics algorithms

As many real-world optimization problems become increasingly complex, better optimization algorithms are always needed. Many problems can be precisely formulated, but they are rather difficult or impossible to solve, either analytically or through conventional numerical procedures. This is the case when the problem is non-convex and, therefore, inherently nonlinear and multimodal. Therefore, in these situations the use of powerful techniques such as heuristic optimization methods is important. Hence, various heuristic optimization methods have been developed to improve solutions to optimization problems.

Heuristics are methods which investigate desirable (near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimal a particular feasible solution is [6]. Often the term heuristic is linked to algorithms mimicking some behaviour found in nature, the principle of evolution through selection and mutation (genetic algorithms), the annealing process of melted iron (simulated annealing), the self-organization of ant colonies (ant colony optimization), the Artificial Immune Systems (AIS), the Bacterial Foraging Algorithm (BFA), or Particle Swarm Optimization (PSO), which simulates the behaviour of

flock of birds. From another perspective, optimization heuristics which are sometimes labelled as approximation methods are generally divided into two broad classes, constructive methods (also called greedy algorithms) and local search methods [6, 7, 8]. Greedy algorithms construct the solution in a sequence of locally optimum choices. Local search uses only information about the solutions in the neighbourhood of a current solution and is thus very similar to hill climbing where the choice of a neighbour solution locally maximizes a criterion. Local search methods are generally divided into trajectory methods which work on a single solution and population based methods, where a whole set of solutions is updated simultaneously.

In the first class are found threshold methods and TABU search whereas the second class consists of genetic algorithms, particle swarm method, differential evolution methods and ant colonies. All these local search methods have particular rules for either or both, the choice of a neighbour and the rules for acceptance of a solution. All methods, except TABU search, allow uphill moves accept solutions which are worse than the previous one, in order to escape local minima [7].

All population-based search algorithms provide satisfactory results but there is no heuristic algorithm that could provide a performance superior to others in solving all optimization problems [5]. In other words, an algorithm may solve some problems better and some problems worse than others. Hence, proposing new high performance heuristic algorithms is welcome.

In this paper, we will establish a new population-based search algorithm based on dynamic random population with normal distribution which greedily considers the valuable part of the solution space.

3 Formulation of the optimization problem

In most processes there are some parameters whose minimization or maximization is desirable. Call these parameters "problem target" or "objective of optimization problem". But, by considering the problem more, we will see that these parameters must be changeable and controllable by some known factors, with direct or indirect impact on the problem objective, in the first place.

Now, if we consider the relation between our objective and the controlling factors as a function with (n) degree of freedom (n influential factor on the problem target) we can write as follows:

$$Objective = f(X_1, X_2, \dots, X_n), \tag{1}$$

where $f(.)$ is the above mentioned function and it expresses the relationships between these factors and the objective yielding a specific value for a given set of factors.

So, adjusting the set of these parameters for optimum result is sought. Indeed, the problem faced is an optimization problem with n degrees of freedom, whose solution requires considering an exhaustive search tending to infinity if no constraints are imposed on values the n factors can assume. It is therefore not possible to consider all cases and the optimal solution must be searched using an optimization method guaranteed to reach an extremum hoped to be global. For this purpose, many methods with different speed and precision have been proposed which are classified into two broad classes of gradient and non-gradient based algorithms. The heuristics methods, the category our algorithm belongs to, were briefly introduced in the previous section. In this paper, a new method that carries out a heuristic optimization is proposed in a simple manner, in comparison with other techniques, and is based on an iterative search method using a set of random n-dimensional vectors with normal distribution.

3.1 Our proposed optimization algorithm

In this section, the proposed algorithm is formulated. First, an n-dimensional vector is initialized with zero as "Bias vector",

$$\begin{aligned} &A_{(0)} = [a_{1_{(0)}}, a_{2_{(0)}}, \dots, a_{n_{(0)}}]_{1\times n} \\ &a_{j_{(0)}} = 0 \text{ for } j\epsilon[1,n]. \end{aligned} \tag{2}$$

Then, in the first iteration a $(K+1)$-member set of n-dimensional vectors with random components around the bias is created. Call this set of normal random vectors as "dynamic random population",

$$\begin{aligned} &\psi_{(1)} = \left\{X_{i_{(1)}}\right\}_{i=0}^{K} \\ &X_{i_{(1)}} = [x_{i1_{(1)}}, x_{i2_{(1)}}, \dots, x_{in_{(1)}}] \ , \ i\epsilon[0,K] \\ &x_{ij_{(1)}} \\ &j\epsilon[1,n] = \begin{cases} a_{j_{(0)}} + \mu\Omega i\epsilon[1,K] \ , \Omega\sim N(0,\zeta) \\ a_{j_{(0)}} \qquad\qquad\qquad\quad i = 0 \end{cases} , \end{aligned} \tag{3}$$

where $\psi_{(1)}$ is the chosen set of random vectors formed by individual vectors $X_{i_{(1)}}$ in the first iteration. Also $a_{j_{(0)}}$ refers to the bias vector components and μ is a constant value used weight the random term. Meanwhile, Ω indicates a zero mean normal random variable with variance ζ.

After completing the first iteration, all members of the chosen $\psi_{(1)}$ are evaluated by Eq. (1) and the best evaluated vector is selected as Eq. (4).

$$\begin{aligned} &X_{\min} = Arg\min_{i\epsilon[0,K]} f(X_i) \\ &Eval_{(1)} = f(X_{\min}), \end{aligned} \tag{4}$$

where $X_{\min}$ and $Eval_{(1)}$ are respectively the best random vector and its evaluation result.

Accordingly, the bias vector is updated as bellow,

$$a_{j_{(1)}} = X_{j_{\min}} \text{for } j\epsilon[1,n]. \tag{5}$$

Therefore, dynamic random population is centred by this new bias vector and the proposed algorithm continues until a certain level of evaluation or a certain number of iterations are reached.

It can be seen that the proposed algorithm only searches the important part of the solution space and gradually approaches the optimum solution. The

flowchart of the proposed algorithm for pre-defined (L) iterations is presented in Fig. 1.

The proposed algorithm with the above mechanism may be trapped in some deep local minimum, thus improving the performance of the algorithm in these cases is necessary. This matter is discussed in the next section.

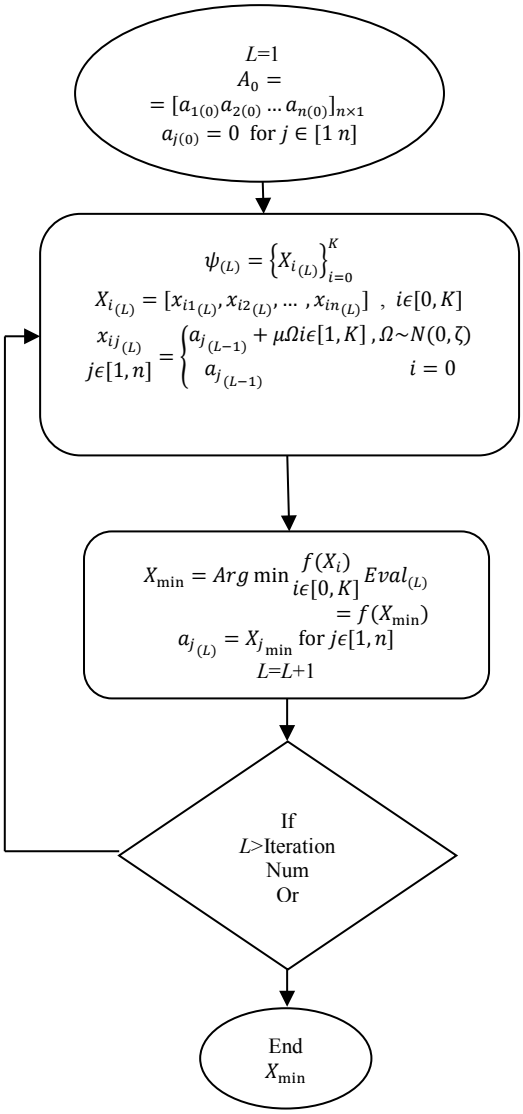

Figure 1 The proposed algorithm flowchart

3.2 Escape mechanism from deep local minimum

With regard to the proposal algorithm being trapped in some deep local minimum, the algorithm is enhanced by adding a good escape mechanism to the updating procedure of the bias vector. The mechanism considered here is as follows:

$$a_{j_{(L)}} = X_{j_{\min_{(L)}}} + \beta a_{j_{(L-1)}},\ j\epsilon[1,n], \tag{6}$$

where β in the above equation is a constant between zero and one which reduces the effect of the previous bias components as bellow. Choose it experimentally β about 10^{-4}.

$$a_{j_{(L+1)}} = X_{j_{\min_{(L+1)}}} + \beta a_{j_{(L)}}$$

$$a_{j_{(L+1)}} = X_{j_{\min_{(L+1)}}} + \beta X_{j_{\min_{(L)}}} + \beta^2 a_{j_{(L-1)}}. \tag{7}$$

3.3 Algorithm accuracy enhancement

Increasing the accuracy of the algorithm is essential for high number of iterations as the result goes closer to the optimum value. For this purpose the variance of the random parts can be reduced as follows:

$$\zeta = \mathrm{e}^{-\frac{n^2}{\rho}}, \tag{8}$$

where ζ, n and ρ in the above equation are respectively the variance of the random variable, the iteration number and an especial constant that controls the variance. In fact, this mechanism reduces the span of search space exponentially based on iteration number.

4 Comparative study and experimental result

Since GA, PSO and ACO algorithms often have good performance, the proposed algorithm is evaluated based on their performances. First, to better understand their performance, the search mechanism of the mentioned algorithms is considered.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented binary as strings of 0 s and 1 s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both

linear chromosomes and trees is explored in gene expression programming. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators. Strength of GAs associates to the parallel nature of their search. A GA is a powerful form of hill climbing algorithm which keeps multiple solutions, eradicates unpromising solutions, and leads to reasonable solutions. All GAs require recombination by virtue of their parents' success. In practice, crossover is the main genetic operator, whereas mutation is used much less frequently. First operator attempts to preserve the beneficial aspects of best solutions and eliminates undesirable parts, while mutation more likely degrades a strong candidate solution than improves it. By limiting the reproduction of weak candidates, GAs eliminate not only that solution but also all of its descendants. This tends to make the algorithm likely to converge towards solutions within a few generations [7].

Another powerful method which is often used is PSO. Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions. PSO is originally attributed to Kennedy, Eberhart and Shi and was first intended for simulating social behaviour as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart describes many philosophical aspects of PSO and swarm intelligence. An extensive survey of PSO applications is made by Poli. PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. PSO can therefore also be used on optimization problems that are partially irregular, noisy, change over time, etc. A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered. Cost function for this method takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution a for which $f(a) \leq f(b)$ for all b in the search-space, which would mean a is the global minimum. Maximization can be performed by considering the function $h = -f$ instead. PSO does not have genetic operators such as crossover and mutation. Searcher agents in PSO (Particles) update themselves with the internal velocity; they also have a memory that is important to the algorithm. Compared with evolutionary programming, evolutionary strategy and genetic programming the information sharing mechanism in PSO is significantly different [10, 13]. In EC approaches, chromosomes share information with each other, thus the whole population moves like one group towards an optimal area but in PSO, only the 'best' particle gives out the information to others. Compared with ECs, all the particles tend to converge to the best solution quickly even in the local version in most cases. Compared to GAs, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust [1, 4, 12].

Meanwhile, ACO is another evolutionary method which is based on ant ability to always find the shortest path between their nest and a food source. In better word, this algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behaviour of ants.

In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food (see Ant communication). Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants following a single path. The idea of the ant colony algorithm is to mimic this behaviour with

"simulated ants" walking around the graph representing the problem to solve.

The original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest.

1. The first ant finds the food source (*F*), via any way (*a*), then returns to the nest (*N*), leaving behind a trail pheromone (*b*).
2. Ants indiscriminately follow four possible ways, but the strengthening of the runway makes it more attractive as the shortest route.
3. Ants take the shortest route; long portions of other ways lose their trail pheromones.

In a series of experiments on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route. A model explaining this behaviour is as follows:

1. An ant (called "blitz") runs more or less at random around the colony;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail of pheromone;
3. These pheromones are attractive; nearby ants will be inclined to follow, more or less directly, the track;
4. Returning to the colony, these ants will strengthen the route;
5. If there are two routes to reach the same food source, then, in a given amount of time, the shorter one will be travelled by more ants than the long route;
6. The short route will be increasingly enhanced, and therefore become more attractive;
7. The long route will eventually disappear because pheromones are volatile;
8. Eventually, all the ants have determined and therefore "chosen" the shortest route.

Ants use the environment as a medium of communication. They exchange information indirectly by depositing pheromones, all detailing the status of their "work". The information exchanged has a local scope, only an ant located where the pheromones were left has a notion of them. This system is called "Stigmergy" and occurs in many social animal societies (it has been studied in the case of the construction of pillars in the nests of termites). The mechanism to solve a problem too complex to be addressed by single ants is a good example of a self-organized system. This system is based on positive feedback (the deposit of pheromone attracts other ants that will strengthen it themselves) and negative (dissipation of the route by evaporation prevents the system from thrashing). Theoretically, if the quantity of pheromone remained the same over time on all edges, no route would be chosen. However, because of feedback, a slight variation on an edge will be amplified and thus allow the choice of an edge. The algorithm will move from an unstable state in which no edge is stronger than another, to a stable state where the route is composed of the strongest edges.

The basic philosophy of the algorithm involves the movement of a colony of ants through the different states of the problem influenced by two local decision policies, viz., trails and attractiveness. Thereby, each such ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. This pheromone information will direct the search of the future ants. Furthermore, the algorithm also includes two more mechanisms, viz., trail evaporation and daemon actions. Trail evaporation reduces all trail values over time thereby avoiding any possibilities of getting stuck in local optima. The daemon actions are used to bias the search process from a non-local perspective [2, 9].

Our proposed algorithm only searches the solution space by generating a random normally distributed population which is concentrated about a targeted bias vector and finds the optimum solution gradually.

In this section, the performance of our proposed algorithm is compared with some of other heuristic search algorithms such as genetic algorithm, particle swarm algorithm and ant colonies algorithm that have relatively good performance. To evaluate the performance of our algorithm, we applied it to 23 standard benchmark functions [5]. These benchmark functions are presented in Section 4.1.

4.1 Benchmark functions

Tabs. 1 ÷ 3 represent the benchmark functions that are often used in experimental studies; we used these benchmark functions as comparison criterion. In these tables, *n* is the dimension of function, *S* is a subset of R^n.

Table 1 Unimodal test functions

Test function	*S*
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]^n$
$F_3(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)$	$[-100,100]^n$
$F_4(X) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100,100]^n$
$F_5(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^n$
$F_6(X) = \sum_{i=1}^{n} ([x_i - 0{,}5])^2$	$[-100,100]^n$
$F_7(X) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	$[-1{,}28; 1{,}28]^n$

Table 2 Multimodal test functions

Test function	*S*
$F_8(X) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500,500]^n$
$F_9(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i + 10)$	$[-5{,}12; 5{,}12]^n$
$F_{10}(X)$ $= -20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$	$[-32,32]^n$
$-\exp\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n} \cos(2\pi x_i)}\right) + 20 + \mathrm{e}$	$[-600,600]^n$ $[-50,50]^n$

$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$	
$F_{12}(X) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i = 1 + \frac{x_i+1}{4}$ $U(x_i,a,k,m) = \begin{cases} k(x_i-a)^m x_i > a \\ 0 \quad -a < x_i < a \\ k(-x_i-a)^m x_i < -a \end{cases}$	$[-50,50]^n$
$F_{13}(X) = 0{,}1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2[1 + \sin^2(3\pi x_i+1)] + (x_n-1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	

Table 3 Multimodal test functions with fixed dimension

Test function	S
$F_{14}(X) = (\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$	$[-65{,}53; 65{,}53]^2$
$F_{15}(X) = \sum_{i=1}^{11}[a_i^2 - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	$[-5{,}5]^4$
$F_{16}(X) = 4x_1^2 - 2{,}1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5{,}5]^2$
$F_{17}(X) = (x_2 - \frac{5{,}1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5,10] \times [0,15]$ $[-5{,}5]^2$
$F_{18}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[0,1]^3$ $[0,1]^6$
$F_{19}(X) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	$[0,10]^4$ $[0,10]^4$ $[0,10]^4$
$F_{20}(X) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	
$F_{21}(X) = -\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	
$F_{22}(X) = -\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	
$F_{23}(X) = -\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	

Functions derived from the work of Yao [5].

We applied our proposed algorithm to these benchmark functions and compared the results with *GA*, *PSO* and *ACO*. In all cases, population size is set to 100. Dimension is 30 and maximum iteration is 100 for functions of Tabs. 1 and 2, and 50 for functions of Tab. 3. Multimodal functions have many local minima and are most difficult to optimize [5]. For multimodal functions, the final results are more important since they reflect the ability of the algorithm in escaping from poor local optima and locating a near-global optimum. So, we included these functions to examine our proposed algorithm performance compared to *GA*, *PSO* and *ACO* that yield good results. Results are not shown here but as an example specific functions results are illustrated bellow. Note that the first result (Fig. 2) does not include the escape mechanism but the latter (Fig. 3) is included, so the next result is much better. But generally, it is clear in these particular cases that the proposed algorithm has the best performance compared to all others (in terms of both accuracy and speed of convergence).

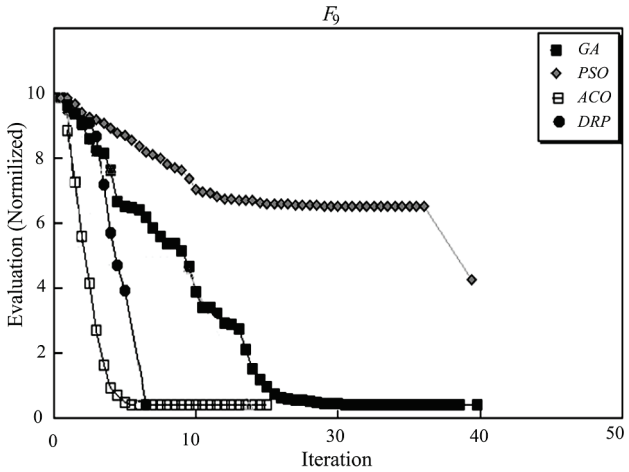

Figure 2 Result of optimization for F_{21} (*DRP* is our proposed algorithm)

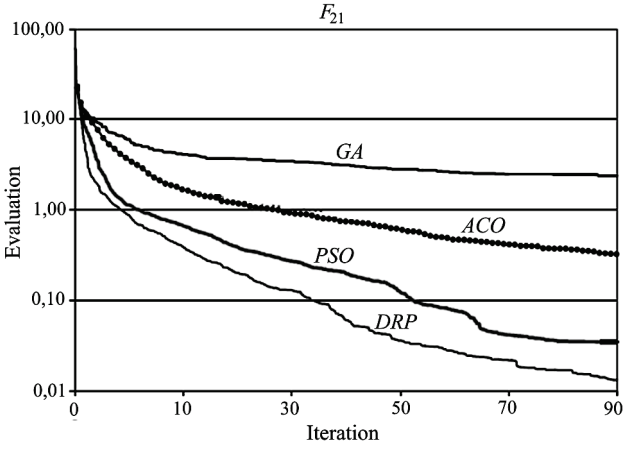

Figure 3 Result of optimization with escape mechanism

Table 4 Minimization result for some mentioned functions

	F_1	F_5	F_6	F_8	F_9	F_{10}	F_{11}	F_{14}	F_{16}	F_{17}
n (dimension)	5	5	5	5	5	5	5	2	2	2
DRP	0	10^{-22}	10^{-19}	−3000	0	10^{-15}	0	1,2	0	0
PSO	1,02	0,02	0,087	−1250	1,07	1,12	0,02	3,12	−1,02	0,4
ACO	5,8	2,56	4,8	−122	41,2	2,54	8,4	2,1	−2,5	0,7
GA	6,2	3,22	5,1	−500	10,4	1,13	0,23	2,3	3,1	0,8

Table 5 Minimization result for some mentioned functions and convergence speed (millisecond)

	F_1	F_5	F_6	F_8	F_9	F_{10}	F_{11}	F_{14}	F_{16}	F_{17}
n (dimension)	5	5	5	5	5	5	5	2	2	2
DRP	0,1	10,2	18,5	18,4	30,8	22,7	36,2	11,2	20,7	14,5
PSO	0,8	12,6	23,8	27,2	29,3	28,4	35,4	15,3	19,5	14,8
ACO	0,5	18,5	18,7	19,8	33,6	26,6	40,1	12,2	21,3	16,3
GA	0,3	15,8	17,1	20,9	24,8	41,1	25,3	13,4	18,7	19,9

In order to evaluate our algorithm, we applied it to a set of various standard benchmark functions. The results obtained by DRP provide superior results in most cases and are comparable to those of PSO, ACO and GA in all cases.

The results of our proposed algorithm and some of other methods (based on some of functions listed in the above tables) are mentioned in Tab. 4 and Tab. 5. These results express once again the superiority of our proposed algorithm from the viewpoint of convergence speed and accuracy.

5 Conclusion

Recently, various Heuristic optimization methods have been developed, that are mostly inspired by nature. In this article, a new optimization algorithm called dynamic random population (DRP) is introduced. DRP is constructed on the basis of searching the most valuable part of the solution space which is normally concentrated about a targeted bias vector.

In order to evaluate our method, we have examined it on a set of various standard benchmark functions. The results obtained by our proposed method in most cases provide superior results and in all cases are comparable with PSO, ACO and GA. Finally it must be noted that our proposed method simplicity may be beneficial for some applications whose calculation cost is valuable.

6 References

[1] Bergh, F. V. D.; Engelbrecht, A. P. A study of particle swarm optimization particle trajectories. // Information Sciences, 176, 8(2006), pp. 937-971.

[2] Dorigo, M.; Maniezzo, V.; Colorni, A. The ant system: optimization by a colony of cooperating agents. // IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26, 1(1996), pp. 29-41.

[3] Gazi, V.; Passino, K. M. Stability analysis of social foraging swarms. // IEEE Transactions on Systems, Man, and Cybernetics – Part B, 34, 1(2004), pp. 539-557.

[4] Wolpert, D. H.; Macready, W. G. No free lunch theorems for optimization. // IEEE Transactions on Evolutionary Computation, 1, 1(1997), pp. 67-82.

[5] Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. // IEEE Transactions on Evolutionary Computation, 3, 2(1999), pp. 82-102.

[6] Ting, T. O.; Rao, M. V. C.; Loo, C. K. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. // IEEE Transactions on Power System, 21, 1(2006), pp. 411-418.

[7] Engelbrecht, A. Computational Intelligence: An Introduction, John Wiley & Sons, Ltd, England. ISBN 0-470-84870-7, 2007.

[8] Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Žumer, V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. // IEEE Transactions on Evolutionary Computation, 10, 6(2006), pp. 646-657.

[9] Tang, K. S.; Man, K. F.; Kwong, S.; He, Q. Genetic algorithms and their applications. // IEEE Signal Processing Magazine, 13, 6(1996), pp. 22-37.

[10] Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. // IEEE Transactions on Evolutionary Computation, 3, 2(1999), pp. 82-102.

[11] Lazar, A.; Reynolds, R. G. Heuristic knowledge discovery for archaeological data using genetic algorithms and rough sets. // Artificial Intelligence Laboratory, Department of Computer Science, Wayne State University, 2003.

[12] Kim, D. H.; Abraham, A.; Cho, J. H. A hybrid genetic algorithm and bacterial foraging approach for global optimization. // Information Sciences, 177, 18(2007), pp. 3918-3937.

[13] Kim, T. H.; Maruta, I.; Sugie, T. Robust PID controller tuning based on the constrained particle swarm optimization. // Automatica, 44, 4(2008), pp. 1104-1110.

Authors' addresses

Seyyed Meysam Hosseini
Department of Communication Engineering,
Faculty of Electrical Engineering, Shahid Beheshti University,
Evin, 19839, Tehran, Iran
E-mail: seyedmeysam.hosseini@yahoo.com
Tel: +98 21 70762175

Hamid Reza Mirsalari
Shoushtar Branch, Islamic Azad University,
Shoushtar, Iran

Hossein Pourhoudhiary
Khozestan Telecom Corporation, Iran